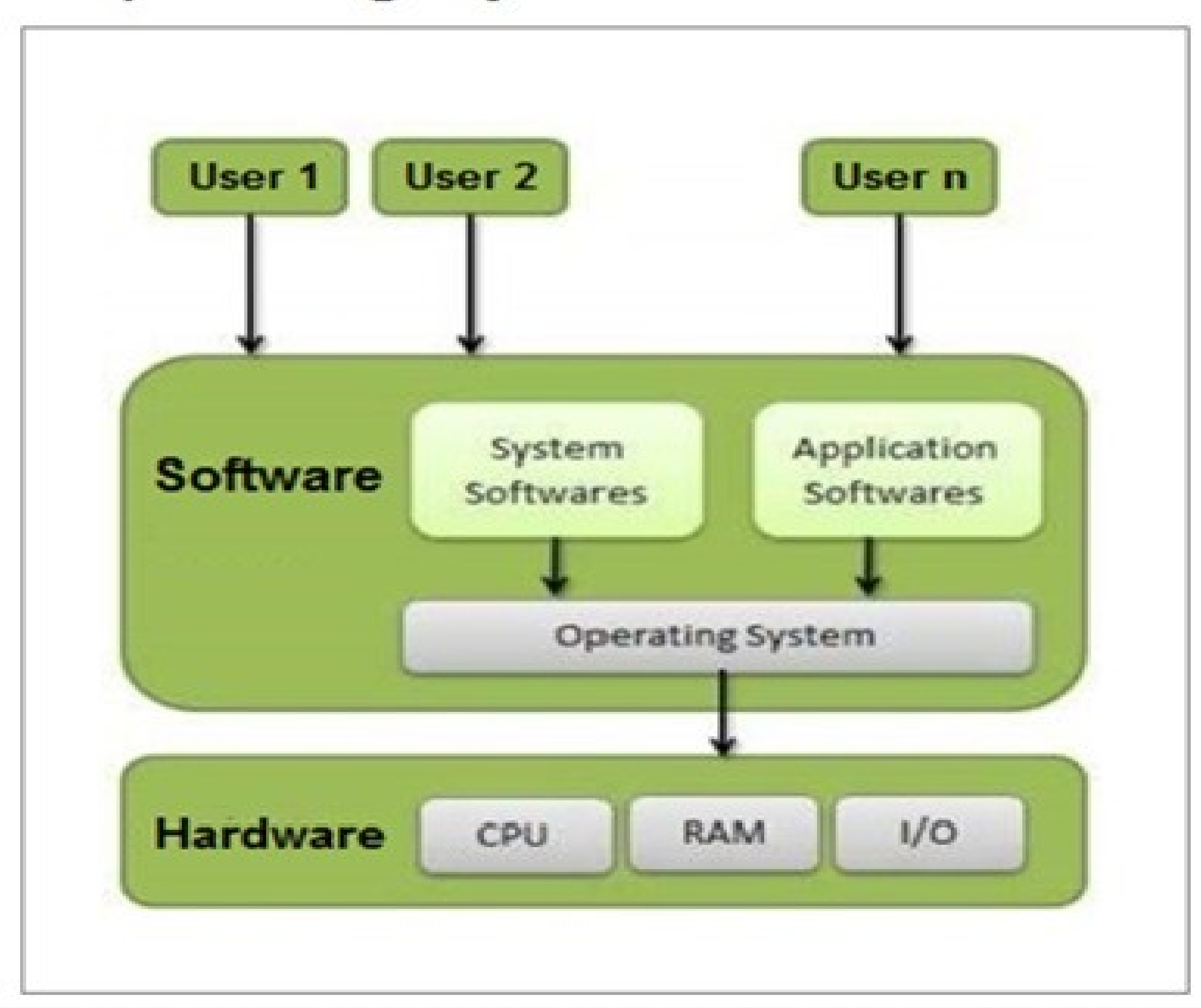
Operating System

An Operating System (OS) is an **interface** between a computer user and computer hardware.

An OS acts as a **Resource Manager**- manages system resources in an unbiased fashion for both hardware and software.

An OS provides a **Platform** on which other application programs are installed.

Operating System Architecture



Goals of OS

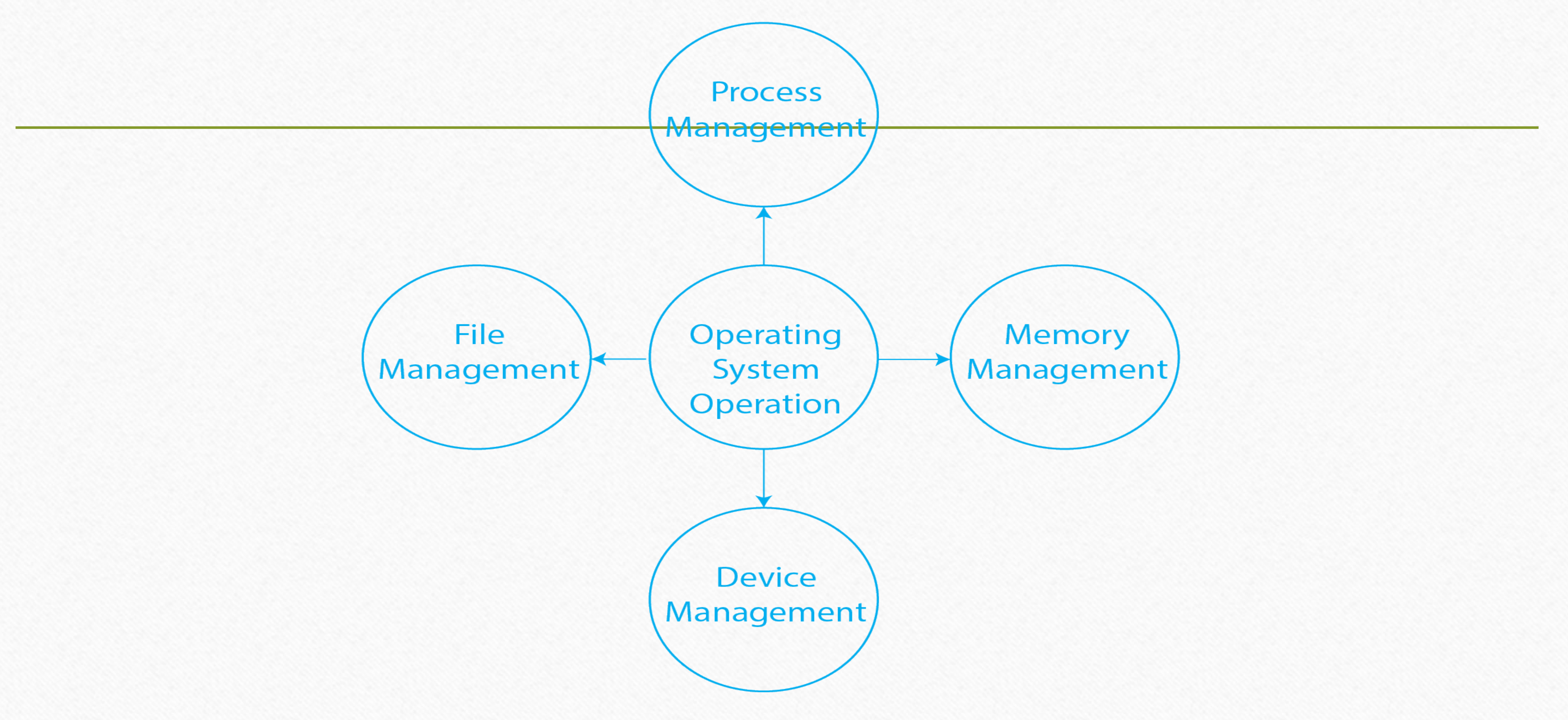
- **Primary Goal**

Convenience/user friendly

- **Secondary Goal**

Efficiency

Functions of OS



Process Management

A process is a program in execution. It is a unit of work within the system. Program is a ***passive entity;*** process is an ***active entity***.

The operating system **is responsible for the following activities** in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for process scheduling
- Providing mechanisms for deadlock handling

Memory Management

The operating system manages the computer's primary memory and provides mechanisms for optimizing memory usage.

Memory management activities

- Keeping track of which parts of memory are currently being used and by whom
- Deciding which processes and data to move into and out of memory
- Allocating and deallocating memory space as needed

File Management

The operating system is responsible for organizing and managing the file system, including:

- Creating and deleting files and directories
- Primitives to manipulate files and directories
- Mapping files onto secondary storage
- Backup files onto stable (non-volatile) storage media

Functions of OS

- **Device Management:** The operating system manages input/output devices such as printers, keyboards, mice, and displays. It provides the necessary drivers and interfaces to enable communication between the devices and the computer.
- **Networking:** The operating system provides networking capabilities such as establishing and managing network connections, handling network protocols, and sharing resources such as printers and files over a network.
- **Security:** The operating system provides a secure environment for the user, applications, and data by implementing security policies and mechanisms such as access controls and encryption.

Historical Evolution :Types of OS

- Batch OS
- Multiprogramming OS
- Multitasking OS
- Multiprocessing OS
- Real Time OS
- Distributed OS

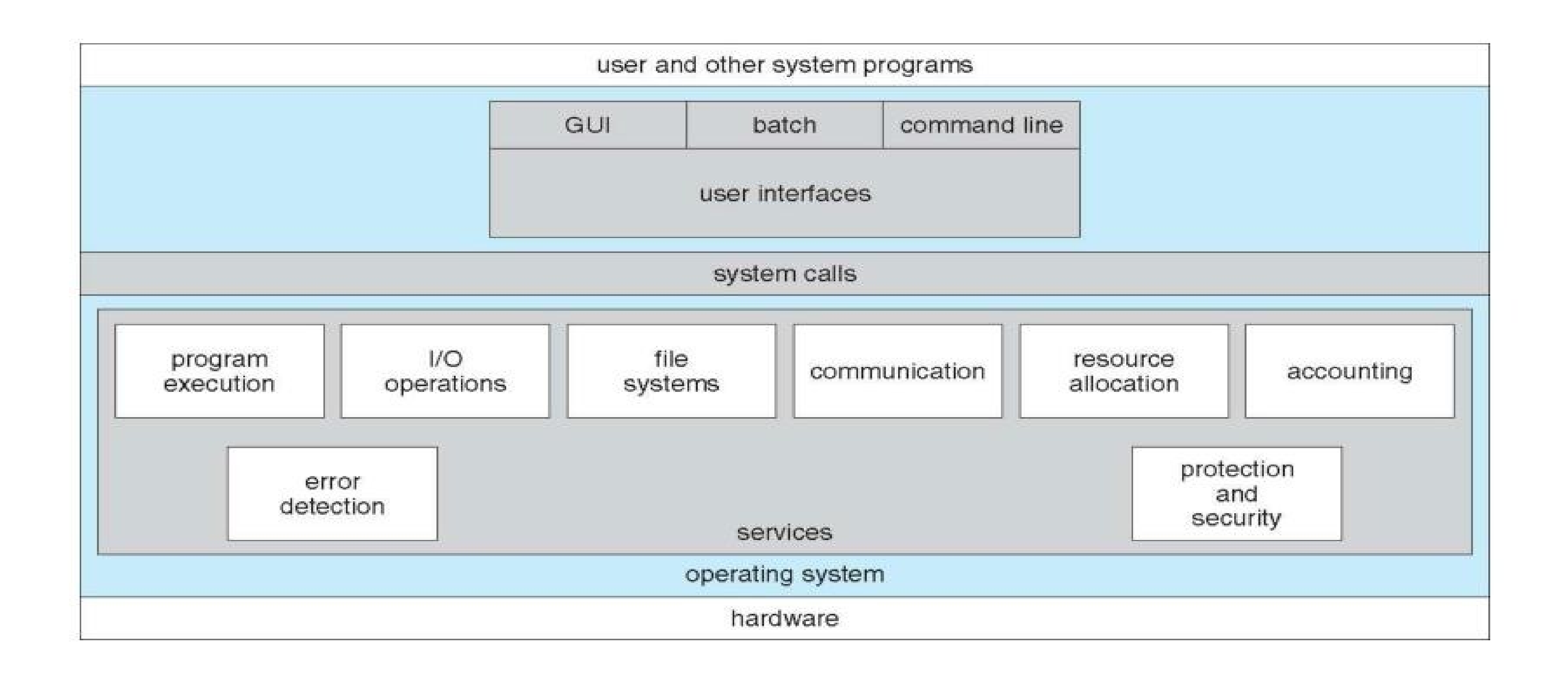
OS Services

- OS **provides environment** for execution of programs.

WHY OS SERVICES REQUIRED

- OS services are provided for the **convenience** of the programmer and to make programming task **easier.**

A View of Operating System Services



OS Services



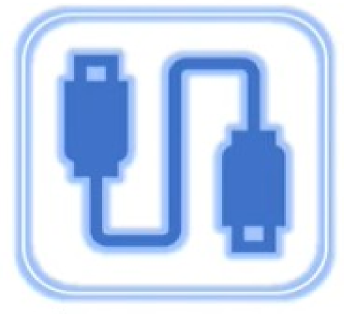
User Interface



Program Execution



File System Manipulation



I/O Operation



Communication



Error Detection



Resource Allocation



Accounting



Protection & Security

User Interface

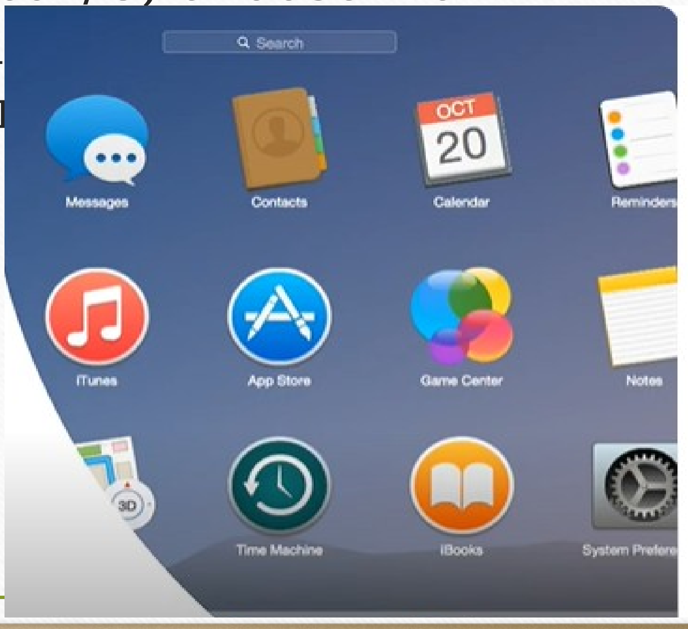
All OS have UI. This can be of following forms:

Command Line Interface

Use text commands and a method for writing them (for ex: a keyboard for typing commands in a specific format within specific opti

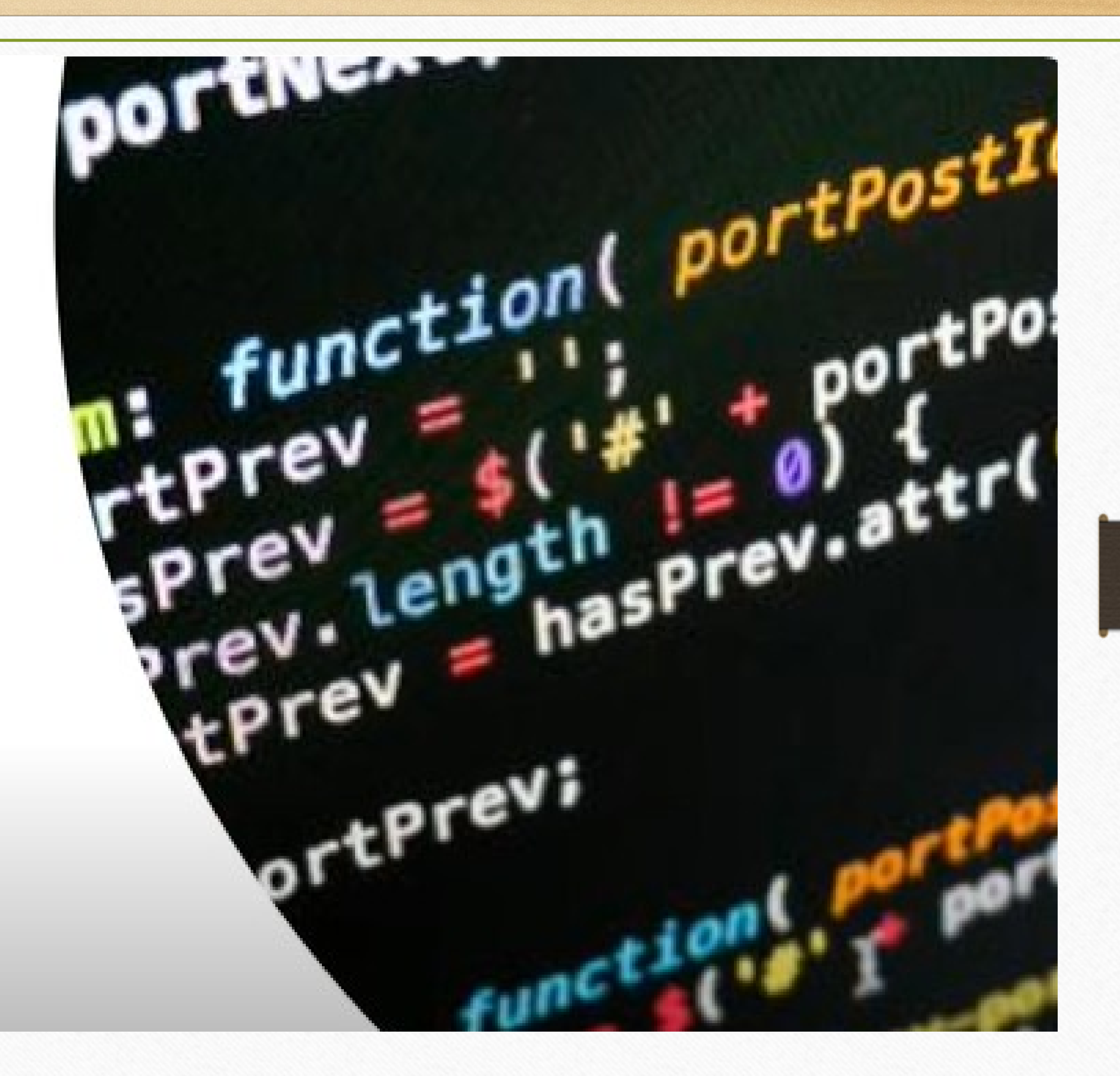
Graphical User Interface

Used most common. In this interface, there is a pointing device to direct I/O, choose from menus, and m keyboard to e



2. Program Execution

- The system must be able to load a program into memory and to run that program.
- The program must be able to end its execution, either normally or abnormally (indicating error).



3. I/O Operations

- A running program may require I/O, which may involve a file or an I/O device.
- For specific devices, special functions may be desired (such as recording to a CD or DVD drive).
- For efficiency and protection, users usually cannot control I/O devices directly. Therefore, the operating system must provide a means to do I/O.



4. File System Manipulation

- Programs need to read and write files and directories. They also need to create and delete them by name, search for a given file, and list file information.
- Finally, some operating systems include permissions management to allow or deny access to files or directories based on file ownership.



5. Communications

- There are many circumstances in which one process needs to exchange information with another process.
- Such communication may occur between processes that are executing on the same computer or between processes that are executing on different computer systems tied together by a computer network.
- Communications may be implemented via **shared memory** or **message passing**.



6. Error Detection

- The operating system needs to be detecting and correcting errors constantly.
- Errors may occur in the CPU and memory hardware, in I/O devices, and in the user program (such as an arithmetic overflow, an attempt to access an illegal memory location).
- For each type of error, the operating system should take the appropriate action to ensure correct and consistent computing.
- Sometimes, OS has no choice but to halt the system. At other times, it might terminate the process or return an error code to a process to detect and possibly correct.



7. Resource allocation

- The operating system manages many different types of resources. Some may have special allocation code, whereas others may have much more general request and release code.
- There may also be routines to allocate printers, USB storage drives, and other peripheral devices.



8. Accounting

- This record keeping may be used for accounting (so that users can be billed) or simply for accumulating usage statistics.



9. Protection & Security

- Protection involves ensuring that all access to system resources is controlled.
- Security of the system from outsiders is also important. Such security starts with requiring each user to authenticate himself or herself to the system, usually by means of a password, to gain access to system resources.
- If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.



System Calls

- Provide an interface to the services made available by an OS.
- The program requests several services, and the OS responds by invoking a series of system calls to satisfy the request.
- A system call can be written in assembly language or a high-level language.

Categories of System Calls

System calls can be grouped into five major categories as follows.

- Process control
- File management.
- Device management
- Information Maintenance and
- Communication.

Process control

- End, abort
- Load, execute
- Create process, terminate process
- Get process, terminate process
- Wait for time
- Allocate and free memory

File management.

Some system calls under file management are:

- Create file, delete file
- Open , close
- Read, write, reposition.
- Get file attributes, set fil attributes

Device management

Some system calls under device management are:

- Request Device, release device
- Read, write, reposition.
- Get device attributes and set device attributes
- Logically attach or detach devices

Information Maintenance

- Get time or date, Set time of date
- Logically attach or detach devices
- Information maintenance
- Get system data, Set Systems data
- Get process, file of device attributes
- Set process, file or device attributes

Communication

- Create, delete communication connection.
- Send, receive messages
- Transfer status information
- Attach or detach remote devices.

System call sequence to copy the contents of one file to another file

source file → destination file

Example System Call Sequence

Acquire input file name
  Write prompt to screen
  Accept input
Acquire output file name
  Write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
Loop
  Read from input file
  Write to output file
Until read fails
Close output file
Write completion message to screen
Terminate normally

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor()	chmod() umask()

C program invoking printf() library call, which calls write() system call

